Длительный рефакторинг в большом проекте: цели, команда и трюки на примере Data Lake Яндекс.Такси

Федор Лаврентьев





Наш протагонист в 2018’Q4

11 лет в IT

- 8 лет в машинном обучении и работе с данными
- 6 лет руководства командами до 25 человек
- Построил +/- десяток продуктов «с нуля»

Партнер в консалтинговом бутике

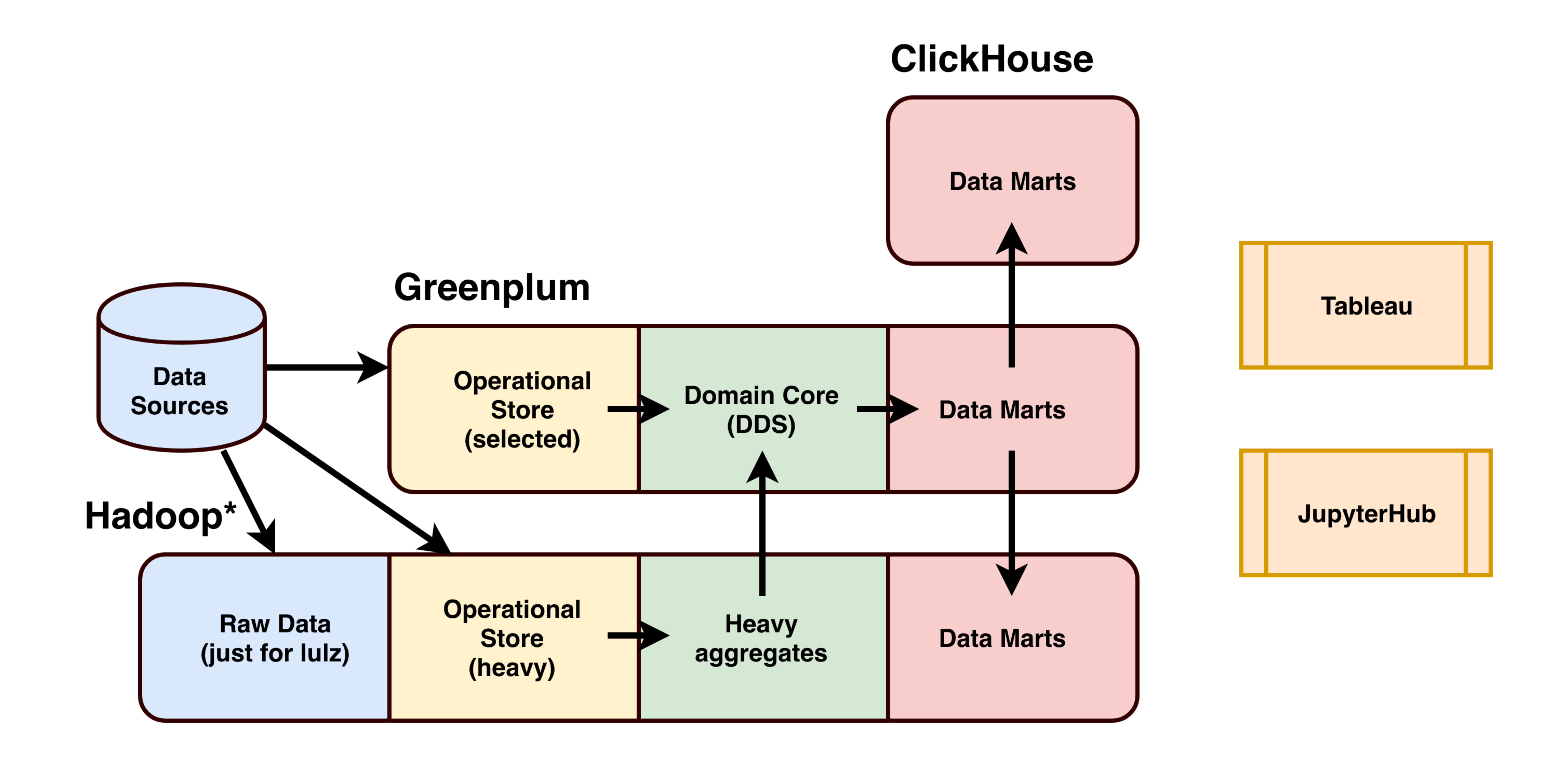
- Занимался внедрением ML-пайплайнов «под ключ»
- Что обычно превращалось в наведение порядка в данных

Получил приглашение в Яндекс.Такси

- Навести порядок в данных



Типичная дата-инфраструктура выглядит так



Федор Лаврентьев, «Граница между Hadoop и MPP», 2019

Data Lake Яндекс.Такси в 2019’Q1

Проект на ~25 человеко-лет

- В разработке с 2016 года
- Команда ~12 человек
- Около 500 ETL-процессов
- Сотни тысяч строк кода
- Несколько типов СУБД

Напряженная обстановка

- Постоянно что-то падает
- Но узнают об этом от пользователей
- Которые истерят в чатиках
- Создаются тысячи мелких тикетов
- Все всем недовольны

Типичная история

- Сначала решали конкретную задачу
- При масштабировании как-то забыли переосмыслить архитектуру
- А теперь уже непонятно как

Давайте это всё перепишем



План рефакторинга



Стой падажжи!

Что происходит на самом деле

Сломалось концептуальное понимание

- Архитектура не выдержала испытания
- Утрачена видимость целевой картины

Бэклог вышел из-под контроля

- Утеряны фокусы
- Сломан time-to-market

Утерян контроль над кодом

- Слишком много независимых процессов
- Написаны в различных подходах

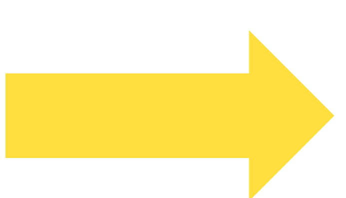
Утерян контроль над рантаймом

- Не все процессы мониторятся
- Есть системно повторяющиеся баги



Понятно, как это чинить

Сломалось концептуальное понимание
- Архитектура не выдержала испытания
- Утрачена видимость целевой картины



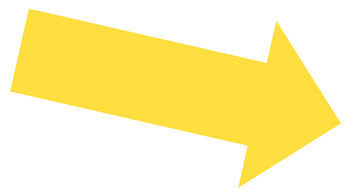
Построить целевую платформу
- Придумать целевую архитектуру
- И план перехода к ней
- Реализовать «фундамент» платформы

Бэклог вышел из-под контроля
- Утеряны фокусы
- Сломан time-to-market



Починить рутинную работу подразделения
- Почистить и приоритизировать бэклог
- Выстроить работу с заказчиками
- Наладить конвейер поставки кода

Утерян контроль над кодом
- Слишком много независимых процессов
- Написаны в различных подходах



Утерян контроль над рантаймом
- Не все процессы мониторятся
- Есть системно повторяющиеся баги

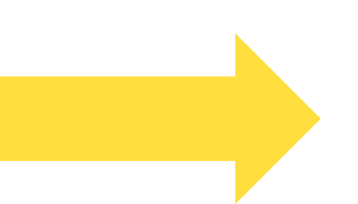


Убрать техдолг из кодовой базы
- Найти и замониторить всё важное
- Переписать легаси-процессы
- Перестать плодить легаси-код



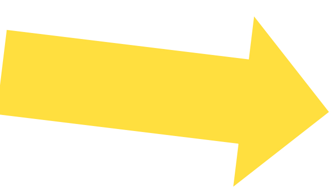
Понятно, как это чинить

Сломалось концептуальное понимание
- Архитектура не выдержала испытания
- Утрачена видимость целевой картины



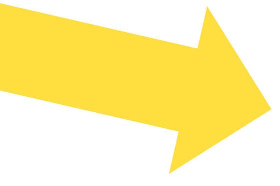
Построить целевую платформу
- Придумать целевую архитектуру
- И план перехода к ней
- Реализовать «фундамент» платформы

Бэклог вышел из-под контроля
- Утеряны фокусы
- Сломан time-to-market

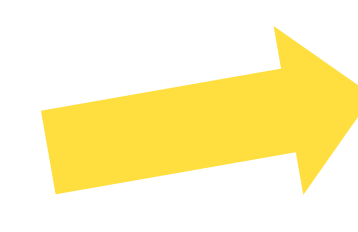


Починить рутинную работу подразделения
- Почистить и приоритизировать бэклог
- Выстроить работу с заказчиками
- Наладить конвейер поставки кода

Утерян контроль над кодом
- Слишком много независимых процессов
- Написаны в различных подходах



Утерян контроль над рантаймом
- Не все процессы мониторятся
- Есть системно повторяющиеся баги



Убрать техдолг из кодовой базы
- Найти и замониторить всё важное
- Переписать легаси-процессы
- Перестать плодить легаси-код



Сначала надо понять, куда мы идём и как

Сломалось концептуальное понимание
- Архитектура не выдержала испытания
- Утрачена видимость целевой картины



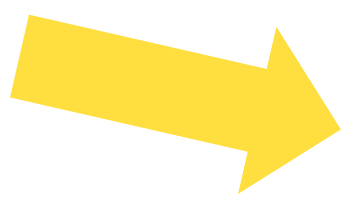
Построить целевую платформу
- Придумать целевую архитектуру
- И план перехода к ней
- Реализовать «фундамент» платформы

Бэклог вышел из-под контроля
- Утеряны фокусы
- Сломан time-to-market



Починить рутинную работу подразделения
- Почистить и приоритизировать бэклог
- Выстроить работу с заказчиками
- Наладить конвейер поставки кода

Утерян контроль над кодом
- Слишком много независимых процессов
- Написаны в различных подходах



Убрать техдолг из кодовой базы
- Найти и замониторить всё важное
- Переписать легаси-процессы
- Перестать плодить легаси-код

Утерян контроль над рантаймом
- Не все процессы мониторятся
- Есть системно повторяющиеся баги





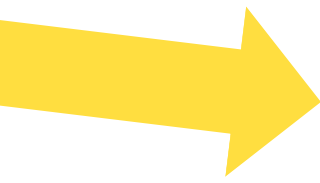
И ещё надо суметь дойти

Сломалось концептуальное понимание
- Архитектура не выдержала испытания
- Утрачена видимость целевой картины



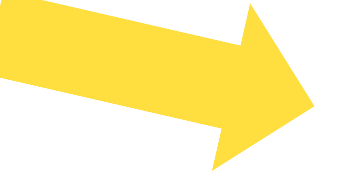
Построить целевую платформу
- Придумать целевую архитектуру
- И план перехода к ней
- Реализовать «фундамент» платформы

Бэклог вышел из-под контроля
- Утеряны фокусы
- Сломан time-to-market



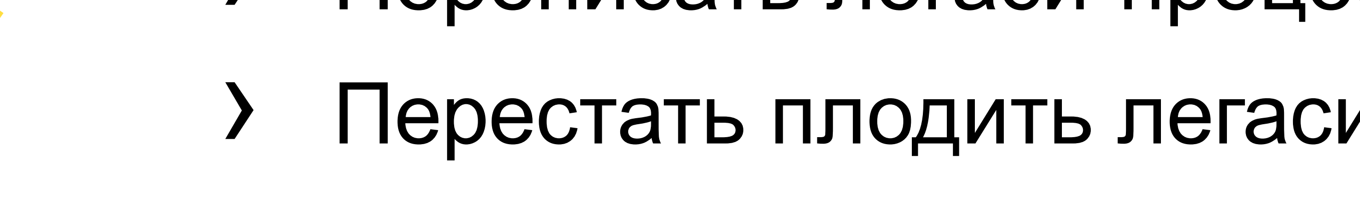
Починить рутинную работу подразделения
- Почистить и приоритизировать бэклог
- Выстроить работу с заказчиками
- Наладить конвейер поставки кода

Утерян контроль над кодом
- Слишком много независимых процессов
- Написаны в различных подходах



Утерян контроль над рантаймом
- Не все процессы мониторятся
- Есть системно повторяющиеся баги

Убрать техдолг из кодовой базы
- Найти и замониторить всё важное
- Переписать легаси-процессы
- Перестать плодить легаси-код





Принцип 1

Знай, куда хочешь прийти

HL HighLoad++ Весна 2021

2019’Q1: Разделение команды на инфраструктуру и продукт

Команда развития инфраструктуры

- Разработчики с широким кругозором
- 60% от стаффа
- Найм маленький

Целеполагание

- Задача – построить новую платформу
- Это чисто инфраструктурные инвестиции
- Команда не делает продуктовых задач
- На бэклог влияет только техлид/архитектор

Команда развития продукта

- Разработчики, близкие к бизнес-задачам
- 40% от стаффа на старте, потом растим
- Почти весь найм сюда

Целеполагание

- Задача – купировать острые боли бизнеса
- Критичные доработки легаси-кода
- Quick win’ы и «репутационные» задачи
- Ничего большого и дорогого



Принцип 2

Явно отделяй инфраструктуру от продукта



2019'Q2: Вдохновляющее многообразие проблем с легаси-кодом

Почему процессы поставки данных падают?

- В источниках данных что-то ломается
- В коде процесса не продуманы краевые случаи
- Другой процесс зааффектил изменениями
- Не хватило ресурсов успеть вовремя
- …

Последствия для бизнеса

- Не приезжают свежие данные
- Данные приезжают не полностью
- Данные начинают расходиться
- Данные теряются

Почему некоторые процессы падают чаще?

- Неустойчивая интеграция с источником
- Большие объемы данных и требования к скорости
- Частые неконтролируемые изменения на стороне источника
- Сложная и недостаточно оттестированная бизнес-логика процесса
- Нетиповые зависимости от других процессов

Давайте это всё перепишем!!



Нельзя просто так взять и переписать сотни тысяч кода

Экономика переписывания

- Нельзя переписывать всё подряд – это заблокирует всю команду продукта
- Если ничего не переписывать – команда продукта будет свободна
- Но проблемы от легаси-кода стоят денег
- Можно оценить стоимость поддержки и оценить целесообразность переписывания

Нельзя просто так взять и переписать сотни тысяч кода

Экономика переписывания

- Нельзя переписывать всё подряд – это заблокирует всю команду продукта
- Если ничего не переписывать – команда продукта будет свободна
- Но проблемы от легаси-кода стоят денег
- Можно оценить стоимость поддержки и оценить целесообразность переписывания

Начнем считать стоимость инцидентов

- Обычно процесс фиксации уже есть, просто сделаем его более формальным
- Для каждого процесса будем считать число падений и время, затраченное на их стабилизацию

Выделим квоты продуктовых команд

- 20-40% времени – на техдолг
- Задачи – упрощение логики процессов, их декомпозиция, стабилизация работы с источником

Дополнительно введем «дежурство»

- Человек, на 100% времени ответственный за стабильность сервиса
- Сам оперативно чинит мелкие баги
- Сложные баги диагностирует и делегирует
- Следит за последствиями релизов
- Дежурят по очереди все разработчики



Трюк 1

Выделенная ресурсная квота

Трюк 2

Дежурство

HL++ HighLoad++ Весна 2021

Как считать стоимость инцидентов

Метаданные процессов

- трекать каждый отдельный запуск каждого процесса (через логи, например)
- Присвоим каждому процессу уникальный ID, каждому запуску – уникальный run ID

Инструменты дежурного

- На основе логов будем собирать информацию о падениях процессов
- Общий дашборд текущего состояния
- Фильтры для отслеживания сбоев в истории
- Кнопочки «Создать тикет», «Перейти к тикетам», «Перейти к статистике» и пр.

Отчетность по падениям

- Пробросим ID процесса в таск-трекере
- Сделаем отчет по тикетам, увидим часто падающие процессы

Расчет стоимости

- Стоимость в рублях считать тяжело
- Легко посчитать стоимость поддержки в человеко-часах
- Можно ввести SLA на аптайм системы
- Для процесса можно отследить месячный вклад в стоимость поддержки и пробои SLA



Принцип 3

Лечи там, где болит

Принцип 4

Знай, где болит

2019’Q3-Q4: Новая платформа оформилась

Определились с архитектурой

- Стек технологий, используемые СУБД
- Архитектура потоков данных
- Стандарт моделирования данных
- Принципы разграничения доступов

Закодили новую платформу

- Единый фреймворк для ETL-процессов
- Типовой деплой, запуск и мониторинг
- Тонны косметической функциональности

HL HighLoad++ Весна 2021

И проблем стало больше

Определились с архитектурой

- Стек технологий, используемые СУБД
- Архитектура потоков данных
- Стандарт моделирования данных
- Принципы разграничения доступов

- Как слезать с «лишних» СУБД?
- Что делать со странными потоками данных?
- Кто должен перемоделировать данные?
- Что делать с существующими доступами

Закодили новую платформу

- Единый фреймворк для ETL-процессов
- Типовой деплой, запуск и мониторинг
- Тонны косметической функциональности

- Кто должен переписывать код под новый фреймворк?
- Что делать со старым кодом?
- Как «продать» в продуктовые команды новую функциональность?



Прекрасно продумать всё заранее

Определились с архитектурой

- Стек технологий, используемые СУБД
- Архитектура потоков данных
- Стандарт моделирования данных
- Принципы разграничения доступов

Закодили новую платформу

- Единый фреймворк для ETL-процессов
- Типовой деплой, запуск и мониторинг
- Тонны косметической функциональности

Жесткий фокус на целевой архитектуре

- Всё новое делается только в целевой архитектуре
- При доработке легаси «докидываем» в задачу перевод в целевую архитектуру
- Есть регламент, что можно доделывать в нецелевой архитектуре

Повсеместные режимы совместимости

- Враппер для старых способов запуска
- Косметические фичи опциональны
- Много статических проверок кода

Некоторые вещи придется сделать сразу

- Типовой деплой, зависимости окружения



Принцип 5

Готовься жить в эпоху перемен



2020’Q1: Код сам себя не рефакторит

Регламент про целевую архитектуру сбоит

- Продуктовые команды привыкли к локальным задачам с быстрым результатом
- Переход из хаоса в формализованную систему воспринимается болезненно
- Команды регулярно наталкиваются на баги в фреймворке и недопродуманные концепции
- Некоторые процессы не получается перевести на целевую архитектуру

Возникает сопротивление

- Со стороны разработчиков – подрыв устоев
- Со стороны бизнеса – рост time-to-market



Слона нужно есть по частям

Регламент про целевую архитектуру сбоит

- Продуктовые команды привыкли к локальным задачам с быстрым результатом
- Переход из хаоса в формализованную систему воспринимается болезненно
- Команды регулярно наталкиваются на баги в фреймворке и недопродуманные концепции
- Некоторые процессы не получается перевести на целевую архитектуру

Возникает сопротивление

- Со стороны разработчиков – подрыв устоев
- Со стороны бизнеса – рост time-to-market

Разбиваем Data Lake на модули

- Распиливание Data Lake на модули аналогично распиливанию монолита на микросервисы
- У каждого модуля свой «стандарт свежести»
- Общее правило – не ухудшай

Дробим большой фронт работ на проекты

- Проект – выдели модуль и зарефактори его
- Для каждой большой фичи – пилотные проекты по внедрению
- Первый «локальный» пилот делает команда инфраструктуры, второй пилот – команда продукта
- Потом стыдим другие модули за их косность



Трюк 3

Отдельный проект

Принцип 6

Ешь слона по частям

2019’Q1 vs 2020’Q1: Накопленные изменения

Изменения в инфраструктуре

- Целевая картина появилась и работает
- Разработчики узко специализированы
- Растет отдаление от команды продукта

Изменения в продукте

- Команда выросла в 2+ раза
- Самое болезненное легаси переписано
- Quick win’ы собраны
- Налажен контакт с бизнес-заказчиками



2019’Q1 vs 2020’Q1: Смена целеполагания

Изменения в инфраструктуре

- Целевая картина появилась и работает
- Разработчики узко специализированы
- Растет отдаление от команды продукта

Изменения в продукте

- Команда выросла в 2+ раза
- Самое болезненное легаси переписано
- Quick win’ы собраны
- Налажен контакт с бизнес-заказчиками

Новый фокус

- Ускорение работы команды продукта
- Ускорение внедрения новых фичей

Новые процессы в продуктовой команде

- Переходим на работу большими эпиками
- В большие эпики подмешиваем рефакторинг
- Активно вовлекаем бизнес-заказчиков в приоритизацию



Трюк 4

Подмешивание

2020’Q2-…: Разработчики любят рефакторинг. Или нет?

Рефакторинг становится рутиной

- Все кейсы изучены и обложены рецептами
- Задача перестаёт быть эпичной
- Люди скучают и хотят в инфраструктуру
- Появляются намёки на кастовость

Учет начинает дорого стоить

- Счет процессов пошел на тысячи
- Часть из них ещё легаси
- Много связей между легаси-процессами



Сдвигаем фокус и переориентируем найм

Рефакторинг становится рутиной

- Все кейсы изучены и обложены рецептами
- Задача перестаёт быть эпичной
- Люди скучают и хотят в инфраструктуру
- Появляются намёки на кастовость

Увеличиваем разнообразие задач

- Не позволяем разработчикам надолго застрять в рефакторинге
- Поощряем запросы бизнеса на технически сложные и нетиповые хотелки
- Нанимаем больше джунов
- Типовой рефакторинг стал почвой для массовых стажировок

Учет начинает дорого стоить

- Счет процессов пошел на тысячи
- Часть из них ещё легаси
- Много связей между легаси-процессами

Вводим метрики техдолга на модуль

- Соответствие модуля целевой архитектуре
- Использование и полезность каждого объекта
- Статические тесты на качество кода



Трюк 5

Широкий фронт

Принцип 7

Не топи людей в рутине

HL HighLoad++ Весна 2021

Подведем итоги



Итоги

Принципы

- Знай, куда хочешь прийти
- Явно отделяй инфраструктуру от продукта
- Лечи там, где болит
- Знай, где болит
- Готовься жить в эпоху перемен
- Ешь слона по частям
- Не топи людей в рутине

Трюки

- Выделенная ресурсная квота
- Дежурство
- Отдельный проект
- Подмешивание
- Широкий фронт



Спасибо!

Федор Лаврентьев, Яндекс Go

